

****

|  |  |
| --- | --- |
| 课程名称： | 信息系统安全 |
| 实验名称： | Race Condition |
| 姓 名： | 王晨皓、谢华洋、宗威旭 |
| 学 号： | 3180103804、3180103576、3180102776 |

2021年 5月 21日

Lab 2：Race Condition

# Purpose and Content实验目的与内容

Purpose

* understand the principle of race condition and vulnerable program
* understand and practice how to exploit the vulnerability and gain the root privilege
* understand several protection schemes used to counter the race-condition attacks

Content

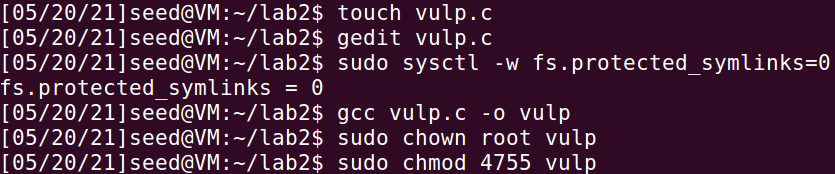
A race condition occurs when multiple processes access and manipulate the same data concurrently, and the outcome of the execution depends on the particular order in which the access takes place. If a privileged program has a race-condition vulnerability, attackers can run a parallel process to race against the privileged program, with an intention to change the behaviors of the program.

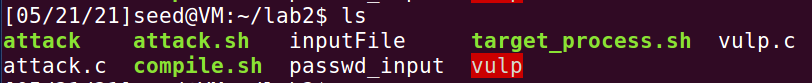
In this lab we will be given a vulnerability program. We need to write a program automate the attack process and an attack program to make use of root privilege. After that we need to test and evaluate several protection schemes that used to counter the race-condition, such as the sticky symlink protection, the Principle of Least Privilege, and so on.

# Detailed Steps 实验过程

Initial Setup & Vulnerable Program

In this step we disable protection about symbol link and prepare a Set-UID vulnerable program for the following attack. Our target is to make use of the time window between access() and fopen() to open different file and then overwrite with the root privilege.





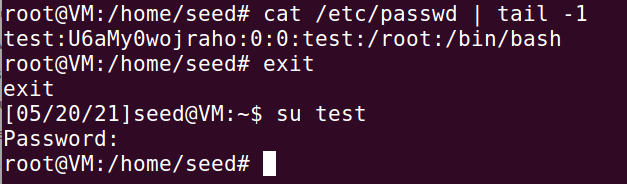
Task 1: Choosing Our Target

We choose to target the password file /etc/passwd which is not writable by normal user. In this step we add the following line to /etc/passwd to add a new user:

test:U6aMy0wojraho:0:0:test:/root:/bin/bash

In the above entry, the first field ‘test’ is username. The second field is hash value and here this value represent for empty password. The third field is UID and UID = 0 means root user.

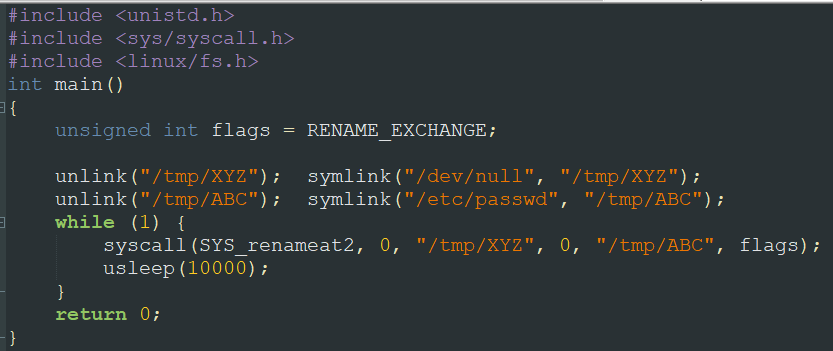
Then try to log into the test account, here the password input is empty. In the following figure we can see that we success to gain root privilege without input password by add edit /etc/passwd file in this way. So we can prepare for the next attack.



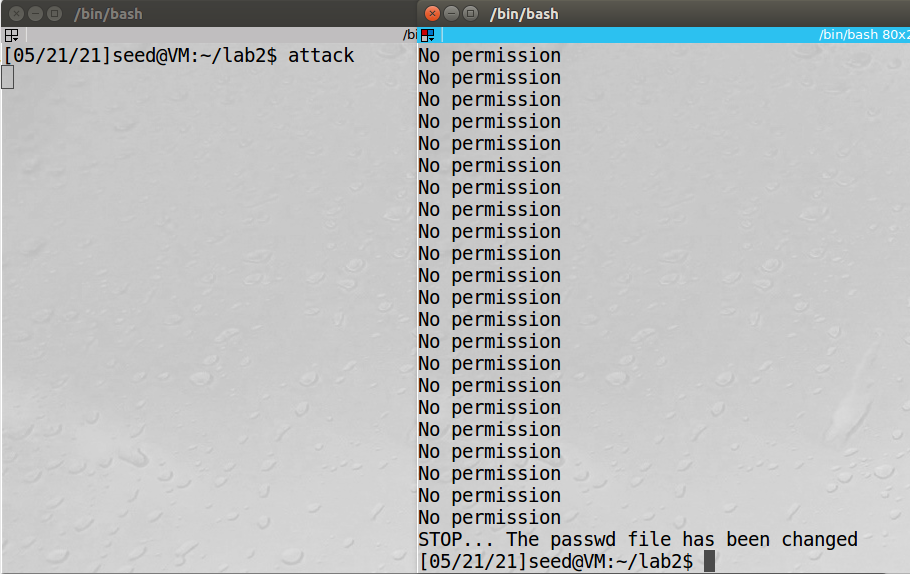
Task 2: Launching the Race Condition Attack

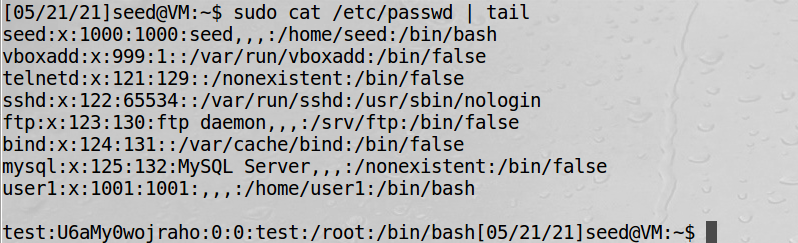
In order to run our attacking program in parallel to race against the target program, we need to write a program to automate the attack process. In our attack program we keep on running the vulnerable program until we succeed to change target file /etc/passwd.

At the same time, we need to attack at the time window between access() and fopen(). So we need run another attack program that keeps on linking it to /etc/passwd.



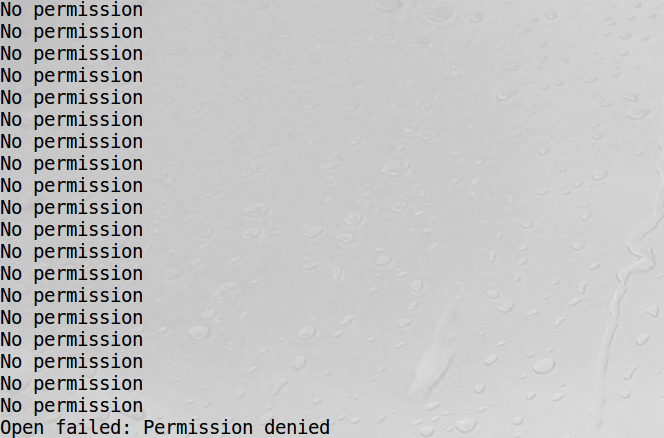
Then, compile and run the attack program and target\_process.sh respectively. We can see the “STOP... The passwd file has been changed” and it suggests we race against the target program, though we can see that the attack is really a probability attack…





Task 3: Countermeasure: Applying the Principle of Least Privilege

Applying the Principle of Least Privilege means we need to withdraw root privilege in time if users don’t need certain privilege. So we add seteuid(getuid()) after access check line to make sure that during the check process, effective user id is your UID. So in the follow figure we can see that the vulnerable program can’t open /etc/passwd because its root privilege has been withdrawn before fopen() excutes.

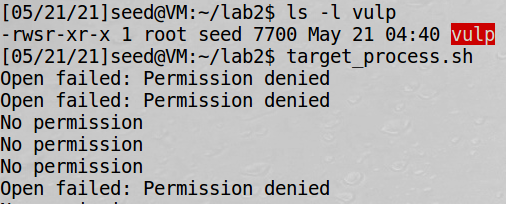


Task 4: Countermeasure: Using Ubuntu’s Built-in Scheme

we can also enable the sticky symlink protection for world-writable sticky directories:

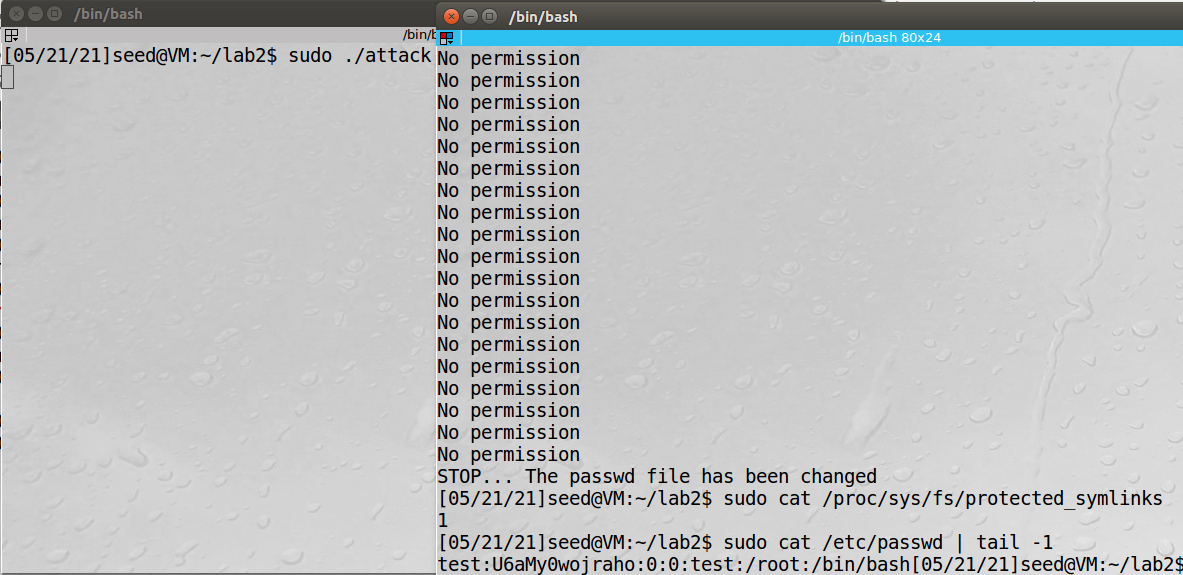
sudo sysctl -w fs.protected\_symlinks=1

Then we conduct our attack and see the following result:



That’s because when the sticky symlink protection is enabled, symbolic links inside a sticky directory (/tmp) can only be followed when the owner of the symlink is either the link follower (EID of the target program) or the directory owner. So as both EID and directory owner are root so normal user(seed) doesn’t have permission to open target file. Because normal user(seed) is symlink owner.

Here is the problem. To attack against sticky symlink protection we must let at least one of the EID and directory owner become same as symlink owner. Thus here if we set the symbol link as root, which means root is symlink owner, we can still make attack effect. We run the attack program as root, and it can still overwrite /etc/passwd.



# Analysis and Conclusion 实验分析与结论

This lab helps us have a better understanding on Race Condition. The vulnerable program seems to have no problem while actually the root privilege still takes effect at the time window between access and fopen. Applying the Principle of Least Privilege keeps program away from unnecessary root privilege. Attacker can’t use memory leak if he doesn’t run his own program either. Sticky symink protection also provides a basic protection which disable the symbol links access when its owner is not equal to EID or directory owner. So attack doesn’t take effect if symbol link’s owner is not root. All in all, Race condition still make use of the time window between command. Withdraw privilege and make critical operation atomic is important to avoid attack from its racer.